CO1212 PRACTICAL WORK ON CO1222

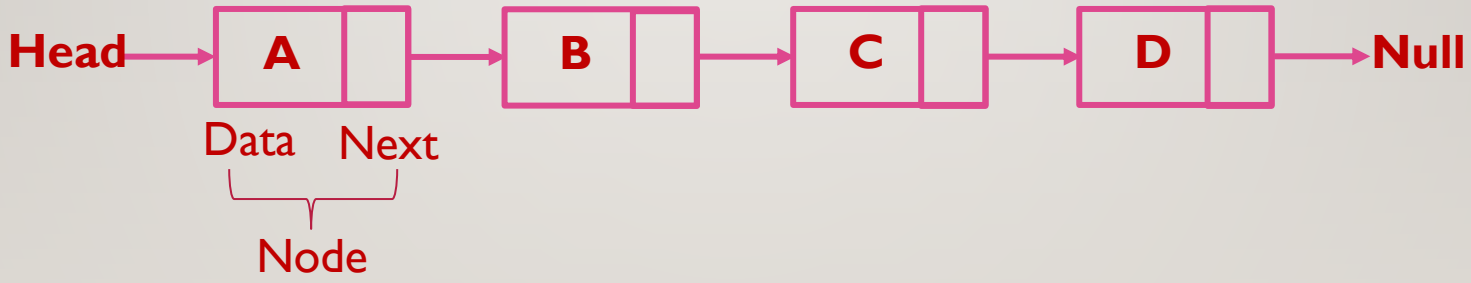
Lesson 12
Linked List

COMPILED BY: W.SRIWATHSAN M.SC (COMPUTER SCIENCE,UCSC)

2 LINKED LIST

Linked List

- A linked list is a linear data structure, in which the elements are not stored at contiguous memory locations. The elements in a linked list are linked using pointers as shown in the below image:



- Linked list consists of nodes where each node contains a data field and a reference(link) to the next node in the list.

3

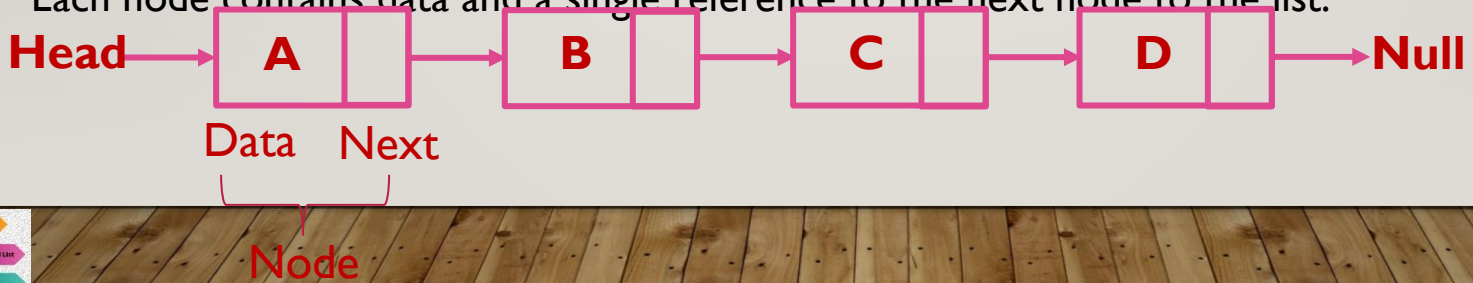
LINKED LIST

Types of Linked List

1. Singly Linked List
2. Doubly Linked List
3. Circular Linked List

1. Singly Linked List

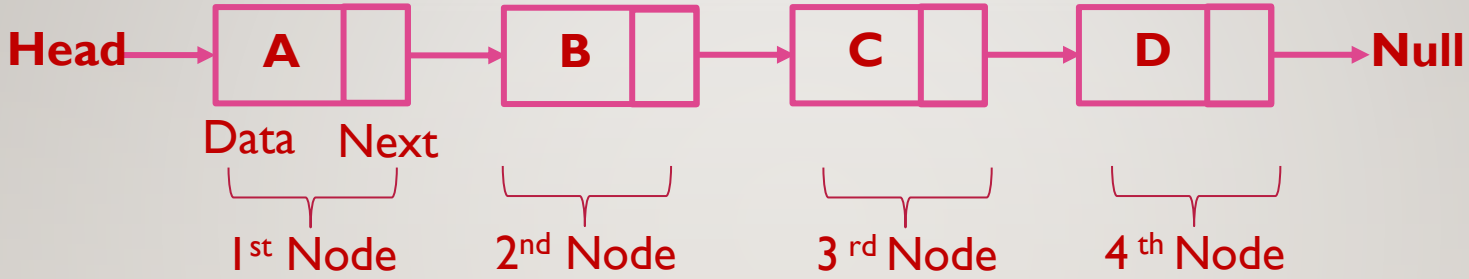
- A Singly linked is a data structure that is made up of list of objects called nodes.
- Each node contains data and a single reference to the next node to the list.



4

LINKED LIST

1. Singly Linked List Cont...



- In the above diagram there are four nodes with some data and a reference to the next node.
- The first note has the reference to the second node and second node has the reference to the third and so on…
- Hear the forth node has the reference to null.
- The first node always has a head as references.

LINKED LIST

Representing a Linked List in Java

- In Java **Linked List** can be represented as a class and a **Node** as a separate class.
- The Linked List class contains a reference of Node class type.

```
// A simple Java program for representing a linked list
class LinkedList {
    Node head; // head of list

    /* Linked list Node. This inner class is made static so that
       main() can access it */
    static class Node {
        char data;
        Node next;
        Node(char d)
        {
            data = d;
            next = null;
        } // Constructor
    }
```

Instance variables

6

LINKED LIST

1. Creating a linked list with four nodes in java

```
// A simple Java program for traversal of a linked list
class LinkedList {
    Node head; // head of list

    /* Linked list Node. This inner class is made static so that
       main() can access it */
    static class Node {
        char data;
        Node next;
        Node(char d)
        {
            data = d;
            next = null;
        } // Constructor
    }
```

7

LINKED LIST

1. Creating a linked list with four nodes in java Cont...

```
/* method to create a simple linked list with 4 nodes*/
    public static void main(String[] args)
    {
        /* Start with the empty list. */
        LinkedList llist = new LinkedList();

        llist.head = new Node('A');
        Node second = new Node('B');
        Node third = new Node('C');
        Node forth= new Node('D');

        llist.head.next = second; // Link first node with the second node
        second.next = third; // Link first node with the second node
        third.next = forth;

  } //End of Class Linked List
}//End of main
```

8

LINKED LIST

2. Traversing a linked list in java.

```
// A simple Java program for traversal of a linked list
class LinkedList {
    Node head; // head of list

    /* Linked list Node. This inner class is made static so that
       main() can access it */
    static class Node {
        char data;
        Node next;
        Node(char d)
        {
            data = d;
            next = null;
        } // Constructor
    }
```

LINKED LIST

2. Traversing a linked list in java Cont...

```
/* This function prints contents of linked list starting from head */
    public void printList()
    {
        Node n = head;
        while (n != null) {
            System.out.print(n.data + " ");
            n = n.next;
        }
    }
```

10

LINKED LIST

2. Traversing a linked list in java Cont...

```
/* This function prints contents of linked list starting from head */
    public void printList()
    {
        Node n = head;
        while (n != null) {
            System.out.print(n.data + " ");
            n = n.next;
        }
    }
```

2. Traversing a linked list in java Cont...

LINKED LIST

```
/* method to create a simple linked list with 4 nodes*/
    public static void main(String[] args)
    {
        /* Start with the empty list. */
        LinkedList llist = new LinkedList();

        llist.head = new Node('A');
        Node second = new Node('B');
        Node third = new Node('C');
        Node forth= new Node('D');

        llist.head.next = second; // Link first node with the second node
        second.next = third; // Link first node with the second node
        third.next = forth;

        llist.printList();
    }
}
```

3. Inserting a node

LINKED LIST

A node can be added in three ways

i. At the front of the linked list
ii. After a given node.
iii. At the end of the linked list.

i. Add a node at the front:

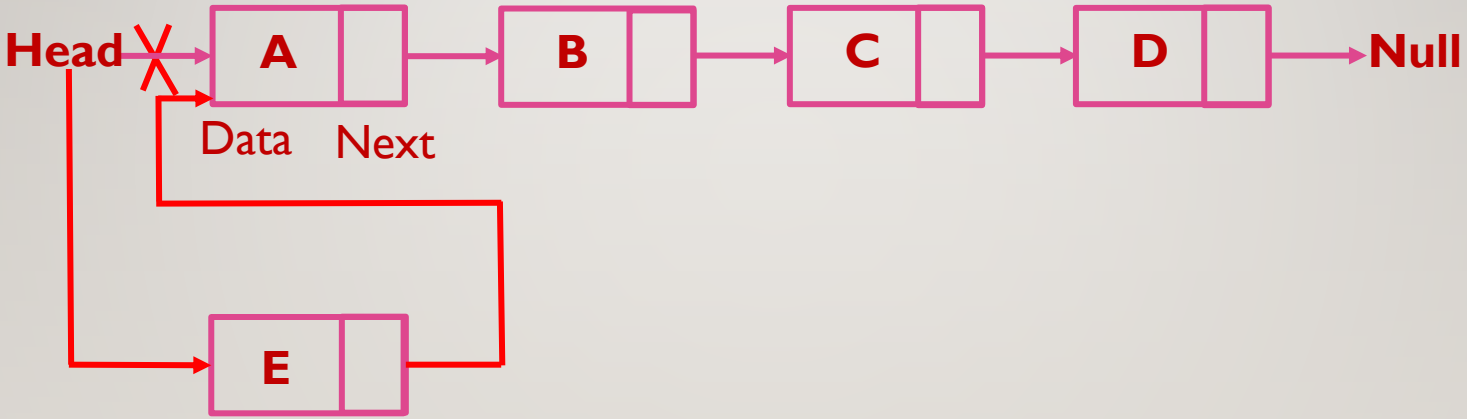
There are four steps involved in this process

1. Allocate the Node
2. Put the data
3. Make next of new Node as head
4. Move the head to point to new Node

3. Inserting a node Cont...

LINKED LIST

i. Add ~~a node at the front cont...~~



14

LINKED LIST

Add a node at the front cont…

```
/* This function is in LinkedList class. Inserts a
   new Node at front of the list. This method is
   defined inside LinkedList class shown earlier */

public void push(char new_data)
{
    /* 1 & 2: Allocate the Node &
              Put in the data*/
    Node new_node = new Node(new_data);

    /* 3. Make next of new Node as head */
    new_node.next = head;

    /* 4. Move the head to point to new Node */
    head = new_node;
}
```

3. Inserting a node

LINKED LIST

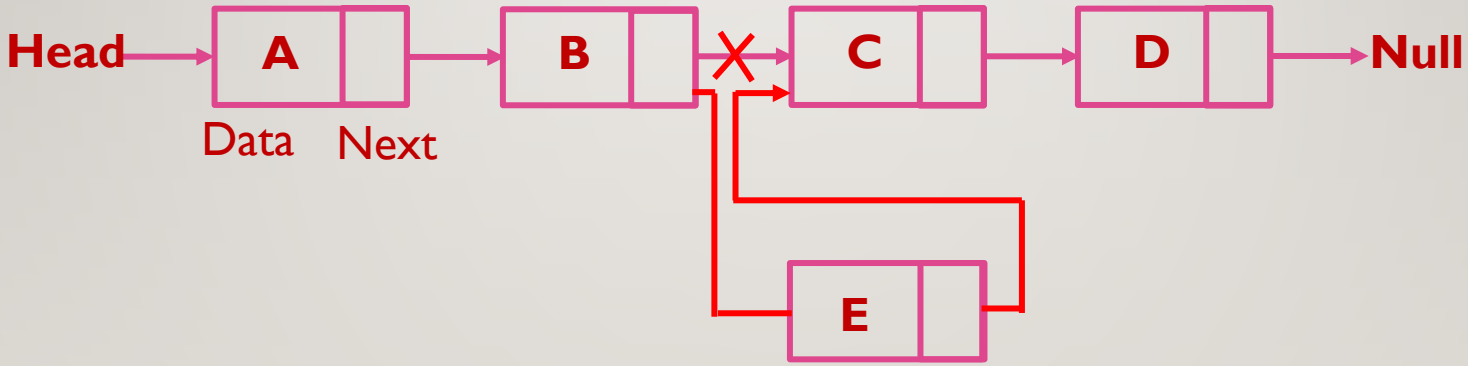
ii. Add a node after a given node:

A pointer is given to a node, and the new node is inserted after the given node.

There are five steps involved in this process

1. Check if the given Node is null
2. Allocate the Node
3. Put in the data
4. Make next of new Node as next of previous node
5. make next of previous node as new node

ii. Add a node after a given node:



ii. Add a node after a given node:

/* This function is in LinkedList class. Inserts a new node after the given prev_node. This method is defined inside LinkedList class shown above */

```
public void insertAfter(Node prev_node, char new_data)
{
    /* 1. Check if the given Node is null */
    if (prev_node == null)
    {
        System.out.println("The given previous node cannot be null");
        return;
    }

    /* 2. Allocate the Node &
    3. Put in the data*/
    Node new_node = new Node(new_data);

    /* 4. Make next of new Node as next of prev_node */
    new_node.next = prev_node.next;

    /* 5. make next of prev_node as new_node */
    prev_node.next = new_node;
}
```

38 Inserting a node

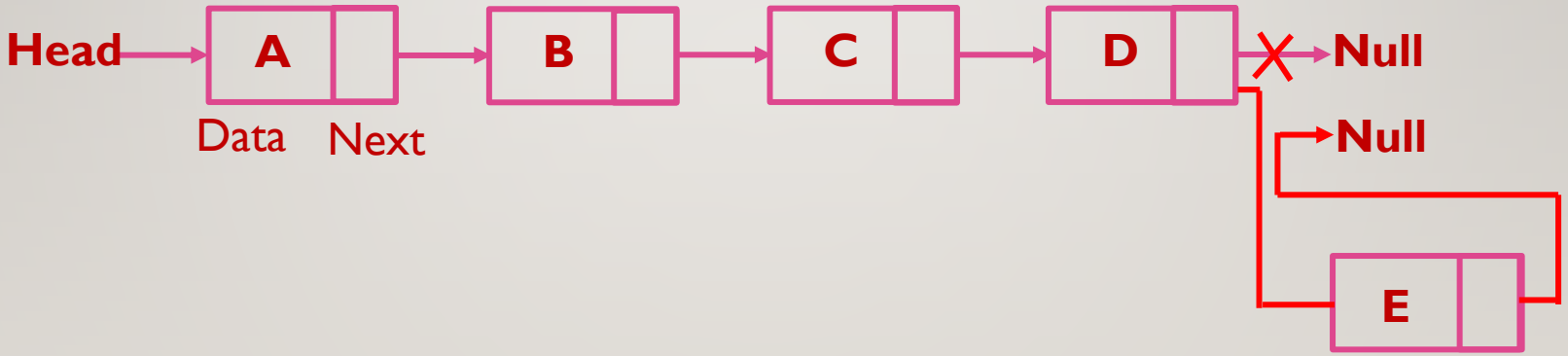
LINKED LIST

iii. Add a node at the end:

There are six steps involved

1. Allocate the Node
2. Put in the data
3. Set next as null
4. If the Linked List is empty, then make the new node as head
   This new node is going to be the last node, so make next of it as null
5. Else traverse till the last node
6. Change the next of last node

iii. Add a node at the end:



20. Inserting a node

LINKED LIST

iii. Add a node at the end:

```
/* Appends a new node at the end.  This method is
   defined inside LinkedList class shown above */
public void append(int new_data)
{
    /* 1. Allocate the Node &
       2. Put in the data
       3. Set next as null */
    Node new_node = new Node(new_data);

    /* 4. If the Linked List is empty, then make the
          new node as head */
    if (head == null)
    {
        head = new Node(new_data);
        return;
    }

    /* 4. This new node is going to be the last node, so
         make next of it as null */
    new_node.next = null;
```

3. Inserting a node

LINKED LIST

iii. Add a node at the end:

```
* 5. Else traverse till the last node */
    Node last = head;
    while (last.next != null)
        last = last.next;

    /* 6. Change the next of last node */
    last.next = new_node;
    return;
}
```